CAIR-EXP: specifications
Version 2.0

Ineui Hong, Eunhye Kim, Ugo Comignani

Supervised by :
Mohand-Said Hacid
Rafiqul Haque

October 27, 2015

Version	Date	Modification
1.0	October 20, 2015	initial version
2.0	October 22, 2015	phase 2 added
3.0	October 27, 2015	Informations about completion of the tasks added

1 Context

This project lies in the context of the semantic web. With the rapid increase of RDF data sources and their ever-growing size, it is important to query disjoint sources more efficiently. One way to achieve this goal is to split the query into smallest sub-queries which are dispatched locally to job nodes. This allows to delegate the jobs of querying the RDF endpoints to the job nodes.

The basis of this project is a software called iSeeker, which permits to query multiple endpoints with the Sparql language. Given a query as input, iSeeker parses and splits it to create one sub-query for each predicate for that input query. Then, these sub-queries are dispatched to the job nodes. When job nodes receive responses from the endpoints, they reduces them before sending the final outcome to the preprocessing node. Finally, the preprocessing node present the answer of the query.

In this context, the goal of this project is to implement iSeeker using new methods to split the query and aggregate the results. Since this is a research oriented of project, only the first phases project are known.

2 Tools and methods

The languages used are java and sparql for the back-end, and javascript for the front-end part.

To deploy the system, three VM will be used (one for the preprocessing node, the others for the job nodes).

In order to follow the instruction of the supervisors, each member of the team is assigned to one task of the current phase. The V-model is used during each phase.

3 Planning

3.1 Phase 1

This phase was finished during the first week. The basis implementation of the interface was received in the middle of the rush week, bringing some delay in the connection between iSeeker's new functionalities and the new interface.

3.1.1 task 1

Implementing an automatic annotation process to create bitset which represent, for each predicate, their availability in the endpoints.
Assigned to Eunhye KIM.

3.1.2 task 2

Implementing a hierarchical clustering algorithm to cluster the predicates according to their availability in the endpoints.
Assigned to Ugo COMIGNANI.

3.1.3 task 3

Implementing a web based interface in JavaScript to allow the users to submit their queries to the system.
Assigned to Ineui HONG.

3.2 Phase 2

For now, only task 1 has been begun ($\approx$ 30% of completetion).

3.2.1 task 1

Implementing a query planner. This component needs to plan the queries so that the communications between preprocessing node, job nodes and sources nodes will be optimized.
Assigned to Ugo COMIGNANI.

3.2.2 task 2

Implementing a query dispatcher to dispatch the sub-queries to the job nodes. This query planner is deployed in the preprocessing node.
Assigned to Eunhye KIM.

3.2.3 task 3

Implementing a query aggregator to aggregate the result returned by the sources. The aggregator must underlie a very efficient algorithm and technique so that the aggregation time is optimized.
Assigned to Ineui HONG.